Code Review

How and Why?

What is code review?

Reading code before it gets merged

- You read your own
- Someone else reads yours

Can be a presentation (boo), in person (yay), asynchronous (yay)

Why code review?

Find bugs
Ensure completeness
Improve code
Improve design
Improve solutions
Learn

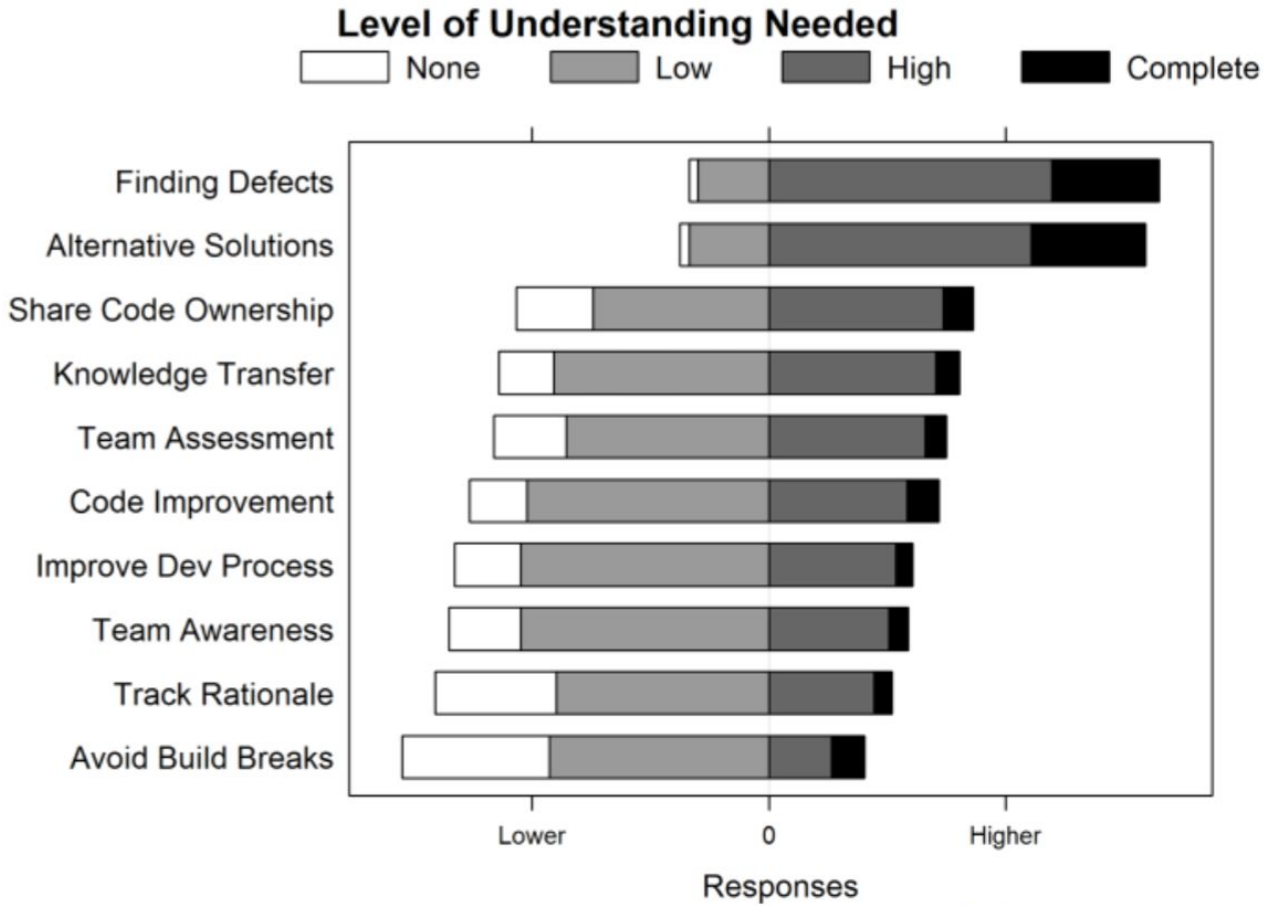

Figure 5. Developers' responses in surveys of the amount of code understanding for code review outcomes.

Align review style with phase of implementation

Good times to ask for review are at 20%, 80% and 99% completion:

- 20% = get design feedback. not so tied to implementation that change is hard
- 80% = get detail feedback. make sure details are right
- 99% = spot-check for last-minute critical errors

If author expects 20%-style feedback and reviewer gives 99%-style ... :sadpanda:

Questions for reviewers to ask themselves

- What kind of feedback is desired?
- What is this trying to do? Why is it trying to do it?
- How would I solve this problem?
- How do we know this implementation solves the problem (testing)?
- How is the design? (4 Rules of Simple Design, SOLID, etc.)
- What should be in this review but is not visible? (minimized diffs, things we forgot to update)

#protip

Avoid ad hominem code review.

- Ask questions.
- Use “we”, not “you” if discussing what to do. E.g.
  - “Why do we need to do X?” vs “You shouldn’t X”

Avoid nit-picking only on code formatting.

Be explicit about which suggested changes are optional vs required.